

ALF USER GUIDE

R. VERFÜRTH

CONTENTS

1. Introduction	1
2. User interface	2
3. Domain and initial grid	3
4. Differential equation	9
5. Discretization	12
6. Solver	12
7. Mesh refinement	14
8. Number of nodes and levels	16
9. Graphics	17
10. Error handling and memory requirement	19

1. INTRODUCTION

ALF is an applet demonstrating **A**daptive **L**inear **F**inite element methods for solving linear stationary convection-diffusion-reaction equations with Dirichlet and Neumann boundary conditions in two-dimensional domains. The finite element partitions consist of triangles and parallelograms; both types may be mixed. The finite element functions are correspondingly linear or bilinear. The partitions are obtained by successively refining an initial partition either uniformly or adaptively. The adaptive refinement is based on residual a posteriori error estimators. The discrete problems are obtained from a standard weak formulation of the differential equation using an SUPG-stabilisation for the convective terms. Multigrid or conjugate gradient algorithms are used for solving the discrete problems. The partitions, finite element solutions, and error indicators can be visualised graphically.

The next section explains ALF's main window. The following six sections show how to use the various choices and text fields that are displayed in the main window and that are used to choose domains, initial grids, differential equations, solvers, error indicators etc. The graphics section explains the possibilities for visualising grids, solutions and error indicators. The last section finally discusses possible error

messages and how to overcome these, in particular when encountering problems with insufficient memory.

2. USER INTERFACE

Starting ALF you'll see a screen (see Fig. 1) which is split into three parts: a left upper part labelled **Input**, a right upper part labelled **Output** and a lower part containing buttons labelled **?**, **Compute**, **Draw Graph**, and **Quit**.

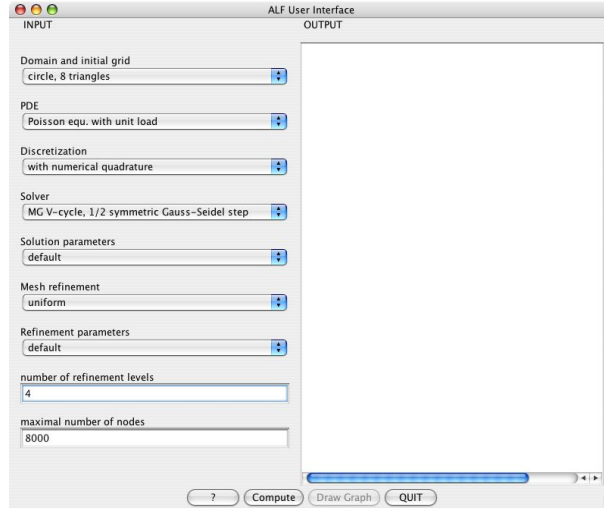


FIGURE 1. ALF's main window

In the *input part* you'll see several choices that allow to choose examples, implemented methods etc. and text fields for setting relevant parameters. The text fields usually exhibit the default values of the corresponding parameters. These choices and text fields will be explained in the following six sections.

Once you have made your choices and entered your favourite parameters you press the **Compute** button to start the computations. The results will be depicted in the *output part*.

Note that some examples, e.g. solving a Poisson equation with a sine-series solution, require additional input. This will be provided via additional windows that show up **after** pressing the **Compute** Button of the main window. These additional windows all have a top label explaining their purpose, one or several labelled text fields for entering the relevant parameters, and an **OK** button. You have to press the latter one after having finished your input in order to continue the computation process.

Once the computation is completed you may eventually want to visualise the results by clicking the **Draw Graph** button. Upon pressing the **Draw Graph** button a graphics window will show up (see Fig. 2).

It is split into three parts: an upper part containing two choices and four text fields labelled **Xmin**, **Xmax**, **Ymin**, and **Ymax**, a lower part containing buttons labelled **Draw** and **Quit**, and a large middle part for the graphics. The values in the text fields determine the drawing area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ and are initially set to default values. *Note* that abscissa and ordinate scale separately, thus a circle will look like an ellipse if $x_{\max} - x_{\min}$ differs from $y_{\max} - y_{\min}$. Clicking the **Draw** button starts the graphics. Clicking the **Quit** button terminates the graphics, closes the graphics window and leads back to the main window.

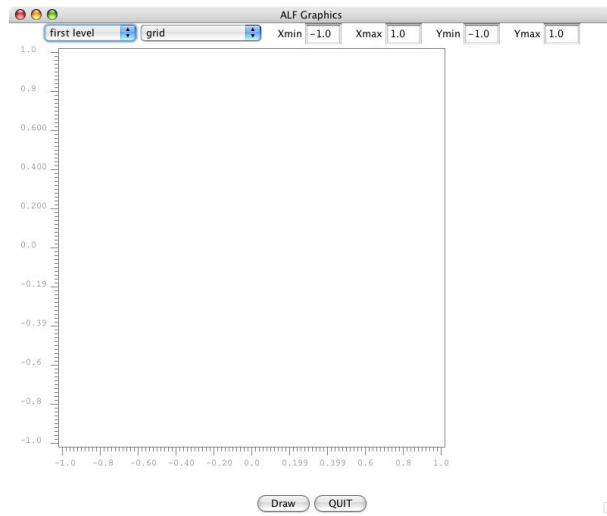


FIGURE 2. ALF's graphics window

You can compute a new set of examples by adjusting the corresponding choices and text fields in the main window and clicking the Compute button.

Upon clicking the ? button in the main window you'll find in the output area a short user guide for ALF.

Clicking the **Quit** button terminates ALF.

3. DOMAIN AND INITIAL GRID

The choice **Domain and initial grid** offers the following options for choosing the domain of the partial differential equation and the coarsest partition for the finite element discretization:

- *Square, 8 triangles*: The domain is the square $[-1, 1] \times [-1, 1]$. The initial partition consists of eight isosceles right-angled triangles with short sides of length 1 and longest side parallel to the line $y = -x$.

- *Square, 4 triangles*: The domain is the square $[-1, 1] \times [-1, 1]$. The initial partition is obtained by drawing the two diagonals of the square.
- *Square, 4 squares*: The domain is the square $[-1, 1] \times [-1, 1]$. The initial partition consists of four squares with sides of length 1.
- *L-shaped, 6 triangles*: The domain and initial partition are obtained by removing the square $[0, 1] \times [-1, 0]$ in the example "Square, 8 triangles".
- *L-shaped, 3 squares*: The domain and initial partition are obtained by removing the square $[0, 1] \times [-1, 0]$ in the example "Square, 4 squares".
- *Circle, 8 triangles*: The domain is the circle with radius 1 and centre at the origin. The initial partition is obtained by replacing the circle by an inscribed regular octahedron and joining its vertices with the origin. When refining an element having an edge with its endpoints on the circle's boundary, the midpoint of the edge will be projected onto the circle. Figures 3 and 4 show for this example the initial partition and the partition resulting from 4 steps of uniform refinement.

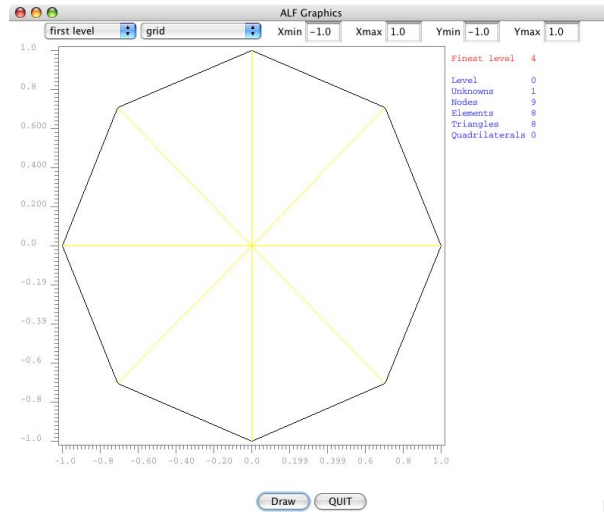


FIGURE 3. Initial partition of example "Circle, 8 triangles"

- *Circle, 4 squares, 8 triangles*: The domain is the circle with radius 1 and centre at the origin. The initial partition is shown in Figure 5. When refining an element having an edge with its endpoints on the circle's boundary, the midpoint of the edge will be projected onto the circle.
- *Segment, 6 triangles*: The domain and initial partition are obtained from those of the example "Circle, 8 triangles" by removing the quarter-segment in the quadrant $x > 0, y < 0$ and the

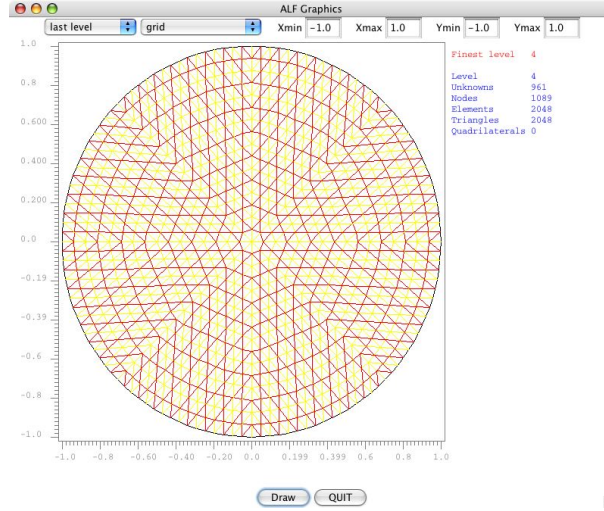


FIGURE 4. Partition resulting from 4 steps of uniform refinement applied to the grid of Fig. 3

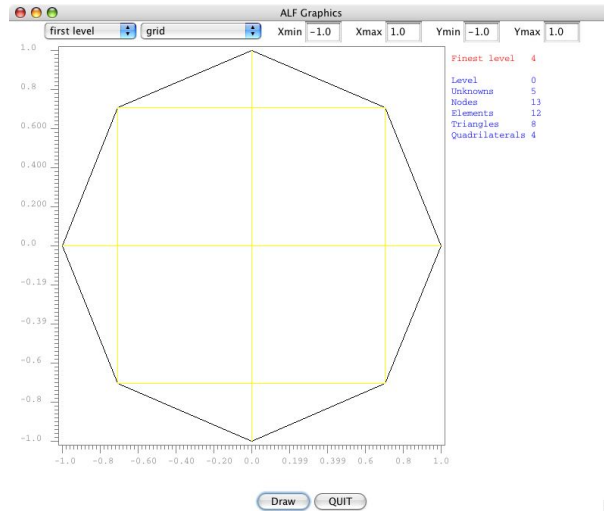


FIGURE 5. Initial partition of the example "Circle, 4 squares, 8 triangles"

corresponding elements. When refining an element having an edge with its endpoints on the circle's boundary, the midpoint of the edge will be projected onto the circle.

- *Segment, 3 squares, 6 triangles*: The domain and initial partition are obtained from those of the example "Circle, 4 squares, 8 triangles" by removing the quarter-segment in the quadrant $x > 0, y < 0$ and the corresponding elements. When refining an element having an edge with its endpoints on the circle's boundary, the midpoint of the edge will be projected onto the circle.

- *Annulus, 12 triangles*: The domain is the annulus, which is formed by concentric circles with radii 0.25 and 1 and centre at the origin. Figures 6 and 7 show for this example the initial partition and the mesh that is obtained after 4 steps of uniform refinement. When refining an element having an edge with its endpoints on the boundary, the midpoint of the edge will be projected onto the corresponding circle.

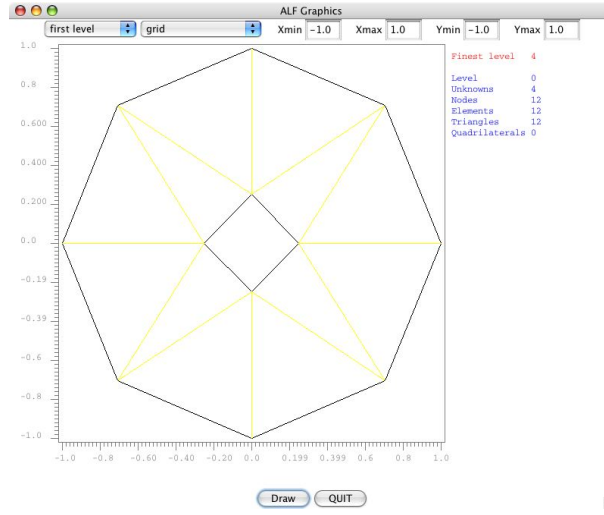


FIGURE 6. Initial partition of example "Annulus, 12 triangles"

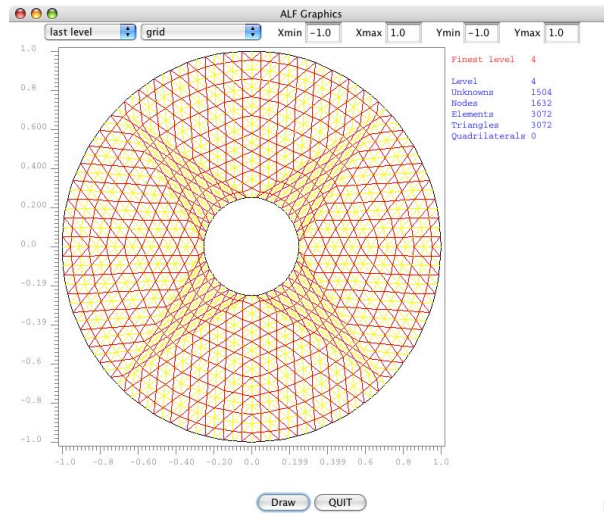


FIGURE 7. Partition resulting from 4 steps of uniform refinement applied to the grid of Fig. 6

- *Double ellipse, 8 triangles*: The domain's boundary consists of two ellipses centred at the origin. The top one has half axes 1

and $\frac{1}{2}$; the lower one has half axes 1 and $\frac{1}{4}$. This domain can be obtained from the circle with radius 1 and centre at the origin by re-scaling the abscissa by a factor $\frac{1}{2}$ and $\frac{1}{4}$ in the upper and lower half-planes respectively. The initial partition is obtained from the one of the example "Circle, 8 triangles" by applying this re-scaling. When refining an element having an edge with its endpoints on the boundary, the midpoint of the edge will be projected onto the corresponding ellipse. Figures 8 and 9 show the initial grid and the mesh obtained after 4 steps of uniform refinement.

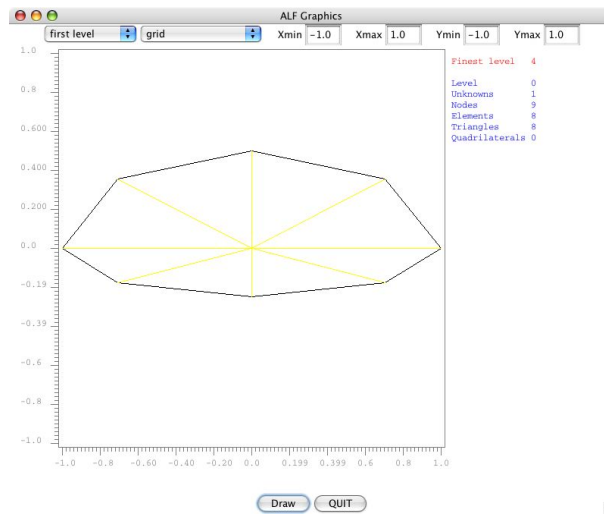


FIGURE 8. Initial partition of example "Double ellipse, 8 triangles"

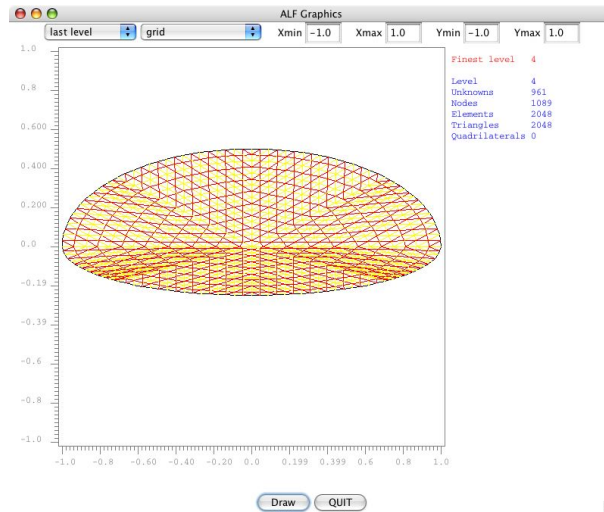


FIGURE 9. Partition resulting from 4 steps of uniform refinement applied to the grid of Fig. 8

- *Double ellipse, 4 squares, 8 triangles*: The domain's boundary consists of two ellipses centred at the origin. The top one has half axes 1 and $\frac{1}{2}$; the lower one has half axes 1 and $\frac{1}{4}$. This domain can be obtained from the circle with radius 1 and centre at the origin by re-scaling the abscissa by a factor $\frac{1}{2}$ and $\frac{1}{4}$ in the upper and lower half-planes respectively. The initial partition is obtained from the one of the example "Circle, 4 squares, 8 triangles" by applying this re-scaling. When refining an element having an edge with its endpoints on the boundary, the midpoint of the edge will be projected onto the corresponding ellipse. Figures 10 and 11 show the initial grid and the mesh obtained after 4 steps of uniform refinement.

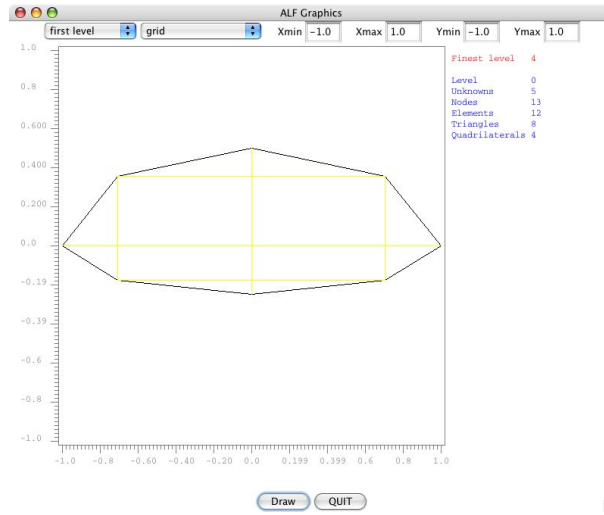


FIGURE 10. Initial partition of example "Double ellipse, 4 squares, 8 triangles"

- *User defined initial grid*: You are first asked to enter the number of nodes. Then you have to enter for each node its x - and y -co-ordinates and a label that specifies whether the point lies in the interior of the domain (label 0), or on a Dirichlet boundary (label > 0), or on a Neumann boundary (label < 0). Then you are asked for the number of elements. Finally you have to enter for each element four integers giving the global numbers of the element's nodes. When the element is a triangle the fourth number must be -1 . The local enumeration of the element vertices must be such that the element interior lies on the left-hand side when passing through the element boundary according to this enumeration.
- *User defined boundary, automatic tessellation*: The domain is split into sub-domains that are each specified by an ordered list

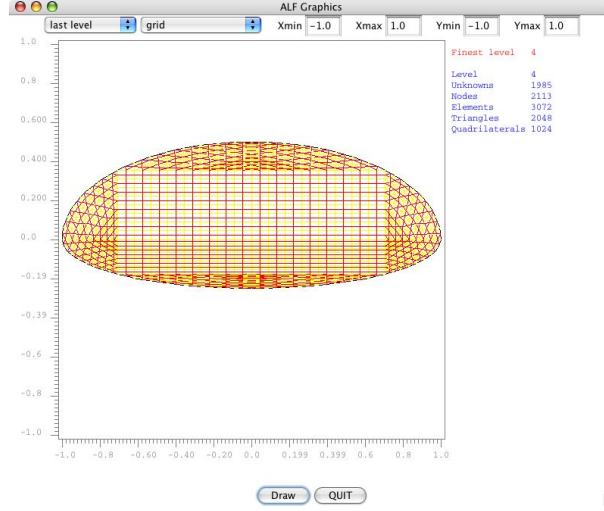


FIGURE 11. Partition resulting from 4 steps of uniform refinement applied to the grid of Fig. 10

of vertices. Each sub-domain is automatically partitioned using a chop-off algorithm. The sub-domains and their partitions are automatically glued together. First you will be asked for the number of sub-domains. For each sub-domain, you then have to specify the number of its vertices. For each vertex you must enter its x - and y -coordinates and a label that specifies whether the point lies in the interior of the domain (label 0), or on a Dirichlet boundary (label > 0), or on a Neumann boundary (label < 0). The vertices of each sub-domain must be ordered such that the interior of the sub-domain lies on the left-hand side when passing through its boundary according to this enumeration.

- *Initial grid of previous example:* The next computation uses the same coarsest partition as in the previous run of ALF. This option is particularly useful when using a user-specified initial partition or when using the automatic-tessellation-facility.

4. DIFFERENTIAL EQUATION

The choice **PDE** provides the following sample differential equations:

- *Poisson equ. with sin-series solution:* The equation is

$$-\Delta u = f$$

with non-homogeneous Dirichlet boundary conditions. The force term f and the boundary data are chosen such that the exact solution is a user-specified sine-series. You will be asked for the number of terms, the amplitudes, and the frequencies.

- *Poisson equ. with cos-series solution*: The equation is

$$-\Delta u = f$$

with non-homogeneous Dirichlet boundary conditions. The force term f and the boundary data are chosen such that the exact solution is a user-specified cosine-series. You will be asked for the number of terms, the amplitudes, and the frequencies.

- *Poisson equ. with bubble solution*: The equation is

$$-\Delta u = f$$

with non-homogeneous Dirichlet boundary conditions. The force term f and the boundary data are chosen such that the exact solution is given by $(1 - x^2)(1 - y^2)$.

- *Poisson equ. with singular solution*: The equation is

$$-\Delta u = f$$

with non-homogeneous Dirichlet boundary conditions. The force term f and the boundary data are chosen such that the exact solution is given by the singular solution for a $\frac{3}{4}$ -segment that takes in polar co-ordinates the form $r^{\frac{2}{3}} \sin(\frac{2}{3}\varphi)$. When using adaptive refinement the mesh should be refined close to the origin.

- *Poisson equ. with unit load*: The equation is

$$-\Delta u = 1$$

with homogeneous Dirichlet boundary conditions. The exact solution is unknown.

- *Reaction-diffusion equation, interior layer*: The differential equation is

$$-\Delta u + bu = f$$

with non-homogeneous Dirichlet boundary conditions. The force term f , the reaction term b , and the boundary data are chosen such that the exact solution is given by $\tanh(\kappa(x^2 + y^2 - \frac{1}{4}))$. You will be asked to enter the parameter κ . The solution exhibits an interior layer along the circle with radius $\frac{1}{2}$ and centre at the origin. When using adaptive refinement the mesh should be refined close to the boundary of this circle.

- *Discontinuous diffusion*: The differential equation is

$$-\operatorname{div}(D \operatorname{grad} u) = 1$$

with homogeneous Dirichlet boundary conditions. The diffusion matrix D is piecewise constant: in the region $4x^2 + 16y^2 < 1$ its diagonal and off-diagonal elements are κ and $\frac{\kappa(\kappa-1)}{\kappa+1}$ respectively; outside this region it is the identity matrix. You will be asked to enter the parameter κ . The exact solution of this problem is unknown, but it is known that it exhibits an interior layer at

the discontinuity of the diffusion matrix. When using adaptive refinement the mesh should be refined close to this region.

- *Convection-diffusion, interior layer*: The differential equation is

$$-\varepsilon \Delta u + \mathbf{a} \cdot \nabla u + bu = f$$

with non-homogeneous Dirichlet boundary conditions. The convection is given by $\mathbf{a}(x, y) = (-y, x)^T$. The force term f , the reaction term b , and the boundary data are chosen such that the exact solution is given by $\tanh(\varepsilon^{-\frac{1}{2}}(x^2 + y^2 - \frac{1}{4}))$. You will be asked to enter the parameter ε . The solution exhibits an interior layer along the circle with radius $\frac{1}{2}$ and centre at the origin. When using adaptive refinement the mesh should be refined close to the boundary of this circle.

- *Convection-diffusion, interior & boundary layer*: The differential equation is

$$-\varepsilon \Delta u + \mathbf{a} \cdot \nabla u + bu = 0$$

in the square $[-1, 1] \times [-1, 1]$ with non-homogeneous Dirichlet boundary conditions. The convection is given by $\mathbf{a}(x, y) = (2, 1)^T$; the boundary condition is 0 at the left and top boundary of the square and 100 at the right and bottom boundary. You will be asked to enter the parameter ε . The exact solution of this problem is unknown. But it is known that it exhibits an exponential boundary layer at the boundary $x = 1, y > 0$ and a parabolic interior layer along the line connecting the points $(-1, 1)$ and $(1, 0)$. When using adaptive refinement the mesh should be refined close to these regions. Since the exponential layer is far stronger than the parabolic layer the refinement will first take place close to the region $x = 1, y > 0$.

- *Convection-diffusion, 3 boundary layers*: The differential equation is

$$-\varepsilon \Delta u + \mathbf{a} \cdot \nabla u + bu = f$$

in the square $[-1, 1] \times [-1, 1]$ with homogeneous Dirichlet boundary conditions. The convection is given by $\mathbf{a}(x, y) = (0, 1 - x^2)^T$; the force term f takes the form $(1 - x^2)(1 - y^2)$. You will be asked to enter the parameter ε . The exact solution of this problem is unknown. But it is known that it exhibits an exponential boundary layer at the top boundary of the square and two parabolic boundary layers along the left and right boundaries. When using adaptive refinement the mesh should be refined close to these regions. Since the exponential layer is far stronger than the parabolic layer the refinement will first take place close to the top boundary of the square.

5. DISCRETIZATION

The differential equation

$$-\operatorname{div}(D\operatorname{grad}u) + \mathbf{a} \cdot \nabla u + bu = f$$

is discretized by

$$\begin{aligned} & \int_{\Omega} \nabla u \cdot D \nabla v + \mathbf{a} \cdot \nabla uv + buv \\ & + \sum_{K \in \mathcal{T}_h} \delta_K \int_K (-\operatorname{div}(D\operatorname{grad}u) + \mathbf{a} \cdot \nabla u + bu) \mathbf{a} \cdot \nabla v \\ & = \int_{\Omega} f v + \sum_{K \in \mathcal{T}_h} \delta_K \int_K f \mathbf{a} \cdot \nabla v. \end{aligned}$$

The SUPG-stabilisation parameters are given by

$$\delta_K = \frac{h_K}{2\|\mathbf{a}\|_2} \left(\coth\left(\frac{\|\mathbf{a}\|_2 h_K}{2\varepsilon}\right) - \frac{2\varepsilon}{\|\mathbf{a}\|_2 h_K} \right),$$

where ε is the smallest eigenvalue of the diffusion matrix D . Non-homogeneous Dirichlet boundary conditions are taken into account by homogenisation. Non-homogeneous Neumann boundary data g give rise to an additional boundary integral on the right-hand side of the discrete problem.

The choice **Discretization** provides two options for building the discrete problem:

- *With numerical quadrature:* All integrals are evaluated using a midpoint rule.
- *Without numerical quadrature.* All integrals are evaluated exactly subject to the assumption that all functions (u , v , D , $\mathbf{a}$, b , f etc.) are piecewise linear respectively bilinear.

6. SOLVER

The choice **Solver** allows you to choose among the following solution algorithms for the discrete problems:

- *MG V-cycle, 1/2 symmetric Gauss-Seidel step:* One iteration consists of one multigrid V-cycle with one forward Gauss-Seidel sweep for pre-smoothing and one backward Gauss-Seidel sweep for post-smoothing.
- *MG V-cycle, n symmetric Gauss-Seidel steps:* One iteration consists of one multigrid V-cycle with a user-specified number of symmetric Gauss-Seidel sweeps for pre- and post-smoothing. You will be asked for the number of pre- and post-smoothing steps. The number of pre-smoothing steps may differ from the number of post-smoothing steps.

- *MG W-cycle, 1/2 symmetric Gauss-Seidel step*: One iteration consists of one multigrid W-cycle with one forward Gauss-Seidel sweep for pre-smoothing and one backward Gauss-Seidel sweep for post-smoothing.
- *MG W-cycle, n symmetric Gauss-Seidel steps*: One iteration consists of one multigrid W-cycle with a user-specified number of symmetric Gauss-Seidel sweeps for pre- and post-smoothing. You will be asked for the number of pre- and post-smoothing steps. The number of pre-smoothing steps may differ from the number of post-smoothing steps.
- *MG V-cycle, 1 downwind Gauss-Seidel step*: One iteration consists of one multigrid V-cycle with one downwind Gauss-Seidel sweep for pre- and post-smoothing. The unknowns are re-ordered lexicographically with respect to the convection direction such that the stiffness matrix is close to lower diagonal form in the limit case of vanishing diffusion.
- *MG V-cycle, n downwind Gauss-Seidel steps*: One iteration consists of one multigrid V-cycle with a user-specified number of downwind Gauss-Seidel sweeps for pre- and post-smoothing. The unknowns are re-ordered lexicographically with respect to the convection direction such that the stiffness matrix is close to lower diagonal form in the limit case of vanishing diffusion. You will be asked for the number of pre- and post-smoothing steps. The number of pre-smoothing steps may differ from the number of post-smoothing steps.
- *MG W-cycle, 1 downwind Gauss-Seidel step*: One iteration consists of one multigrid W-cycle with one downwind Gauss-Seidel sweep for pre- and post-smoothing. The unknowns are re-ordered lexicographically with respect to the convection direction such that the stiffness matrix is close to lower diagonal form in the limit case of vanishing diffusion.
- *MG W-cycle, n downwind Gauss-Seidel steps*: One iteration consists of one multigrid W-cycle with a user-specified number of downwind Gauss-Seidel sweeps for pre- and post-smoothing. The unknowns are re-ordered lexicographically with respect to the convection direction such that the stiffness matrix is close to lower diagonal form in the limit case of vanishing diffusion. You will be asked for the number of pre- and post-smoothing steps. The number of pre-smoothing steps may differ from the number of post-smoothing steps.
- *MG V-cycle, 1 squared Richardson step*: One iteration consists of one multigrid V-cycle with one Richardson iteration for the squared system as pre- and post-smoothing step.
- *MG V-cycle, n squared Richardson steps*: One iteration consists of one multigrid V-cycle with a user-specified number of

Richardson iterations for the squared system as pre- and post-smoothing step. You will be asked for the number of pre- and post-smoothing steps. The number of pre-smoothing steps may differ from the number of post-smoothing steps.

- *MG W-cycle, 1 squared Richardson step*: One iteration consists of one multigrid W-cycle with one Richardson iteration for the squared system as pre- and post-smoothing step.
- *MG W-cycle, n squared Richardson steps*: One iteration consists of one multigrid W-cycle with a user-specified number of Richardson iterations for the squared system as pre- and post-smoothing step. You will be asked for the number of pre- and post-smoothing steps. The number of pre-smoothing steps may differ from the number of post-smoothing steps.
- *CG*: This is a classical conjugate-gradient algorithm.
- *CG with SSOR preconditioning*: This is a classical preconditioned conjugate-gradient algorithm with one symmetric successive over-relaxation sweep for preconditioning.
- *BiCG-Stab*: This is a stabilised bi-conjugate-gradient algorithm.
- *BiCG-Stab with SSOR preconditioning*: This is a stabilised and preconditioned bi-conjugate-gradient algorithm with one symmetric successive over-relaxation sweep for preconditioning.

All solution algorithms terminate when either *MAXIT* iterations have been performed or when the initial residual, measured in the Euclidean norm, has been reduced by at least a factor *TOL*. The parameter *MAXIT* is set to 10 for the multigrid algorithms with Gauss-Seidel smoothing, to 50 for the multigrid algorithms with squared Richardson smoothing and to 100 for the CG and BiCG algorithms. The parameter *TOL* is set to 0.01. The choice **Solution parameters** gives you the opportunity to modify these parameters by choosing the option *user settings* the standard choice being *default*.

7. MESH REFINEMENT

ALF starts its computations on a partition, which is labelled Level 1 and which is obtained by a uniform refinement of the coarsest mesh labelled Level 0 of the actual example. Having computed the discrete solution on a mesh of Level k the partition of the next level is obtained either by a *uniform refinement* or by an adaptive refinement based on various a posteriori *error indicators* and *refinement strategies*. This process is repeated until the prescribed number of refinement levels is reached or until the allocated storage is exhausted (cf. Sections 8 and 10).

When using a *uniform refinement* every element is refined *regularly*, i.e. the midpoints of edges are connected for triangles and the midpoints of opposite edges are connected for quadrilaterals.

ALF has implemented two types of *residual error indicators*:

- *edge residuals* and
- *edge and element residuals*.

Given the differential equation

$$-\operatorname{div}(D\operatorname{grad}u) + \mathbf{a} \cdot \nabla u + bu = f$$

the *edge residuals* of an element K take the form

$$\eta_K^2 = \sum_{E \subset \partial K} \epsilon^{-\frac{1}{4}} \alpha_E \|[\mathbf{n}_E \cdot D\nabla u_h]_E\|_{L^2(E)}^2.$$

Here, u_h is the solution of the discrete problem, ϵ denotes the geometric mean of the two eigenvalues of the diffusion matrix D evaluated at the element's barycentre, $\mathbf{n}_E$ is a unit vector orthogonal to the edge E , $[\cdot]_E$ denotes the jump across the edge E , and the weight α_E is given by

$$\alpha_E = \min\{h_E \epsilon^{-\frac{1}{2}}, \beta^{-\frac{1}{2}}\},$$

where h_E is the diameter of E and β_E is the maximum of 0 and $b - \frac{1}{2}\operatorname{div}\mathbf{a}$ evaluated at the element's barycentre. Edges on a Dirichlet boundary must be ignored. For edges on a Neumann boundary the term $[\mathbf{n}_E \cdot D\nabla u_h]_E$ must be replaced by $g - \mathbf{n} \cdot D\nabla u_h$ where g is the given Neumann datum and $\mathbf{n}$ denotes the exterior unit normal to Ω .

The *edge and element residuals* of a given element K take the form

$$\begin{aligned} \eta_K^2 &= \alpha_K^2 \|f + \operatorname{div}(D\operatorname{grad}u) - \mathbf{a} \cdot \nabla u - bu\|_{L^2(K)}^2 \\ &\quad + \sum_{E \subset \partial K} \epsilon^{-\frac{1}{4}} \alpha_E \|[\mathbf{n}_E \cdot D\nabla u_h]_E\|_{L^2(E)}^2, \end{aligned}$$

where the weight α_K is defined in the same way as the weight α_E with the edge E replaced by the element K .

ALF has implemented two *refinement strategies*:

- (1) the *maximum strategy* and
- (2) the *equilibration strategy*.

Both strategies assume that an error indicator η_K has been computed for every element of a given partition $\mathcal{T}$. Both first mark a percentage *EXCESS* of the elements with largest η_K for refinement. This set of elements is labelled $\mathcal{M}_0$ the set of unmarked elements is labelled $\mathcal{U}$

In the *maximum strategy* an element K in $\mathcal{U}$ is marked for refinement if and only if

$$\eta_K \geq \text{THRESHOLD} \cdot \max_{K' \in \mathcal{U}} \eta_{K'}.$$

These elements form the set $\mathcal{M}_1$.

In the *equilibration strategy* the set $\mathcal{M}_1$ is constructed by taking all elements of $\mathcal{U}$ with largest η_K such that

$$\sum_{K \in \mathcal{M}_1} \eta_K^2 \geq \text{THRESHOLD} \cdot \sum_{K \in \mathcal{U}} \eta_K^2.$$

The union of the sets $\mathcal{M}_0$ and $\mathcal{M}_1$ is the collection of the elements that are refined regularly. In order to avoid hanging nodes additional

elements are refined green or blue as described in the literature (cf. e.g. Chapter 4 of *R. Verfürth: A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques. Wiley-Teubner 1996* or Chapters III.3 and III.4 of the Lecture Notes *Numerische Behandlung von Differentialgleichungen II (Finite Elemente)*).

The choice **Mesh refinement** allows you to choose among the implemented refinement and error estimation algorithms.

The choice **Refinement parameters** allows you to choose the parameters *EXCESS* and *THRESHOLD* described above and the parameter *FIRSTADAPTIVE* that determines the first level on which an adaptive refinement is performed. The default settings are

- $EXCESS = 0.1$,
- $THRESHOLD = 0.5$,
- $FIRSTADAPTIVE = 1$.

When choosing the option *user settings* you will be asked to enter these three parameters.

Figure 12 shows the mesh of Level 8 obtained for the example "Convection-diffusion, interior & boundary layer" with the parameter $\varepsilon = 0.000001$ and the default settings for *EXCESS*, *THRESHOLD*, and *FIRSTADAPTIVE*. Figure 13 shows the mesh of Level 7 for the same example with the same parameters ε and *THRESHOLD* but with $EXCESS = 0.2$ and $FIRSTADAPTIVE = 2$.

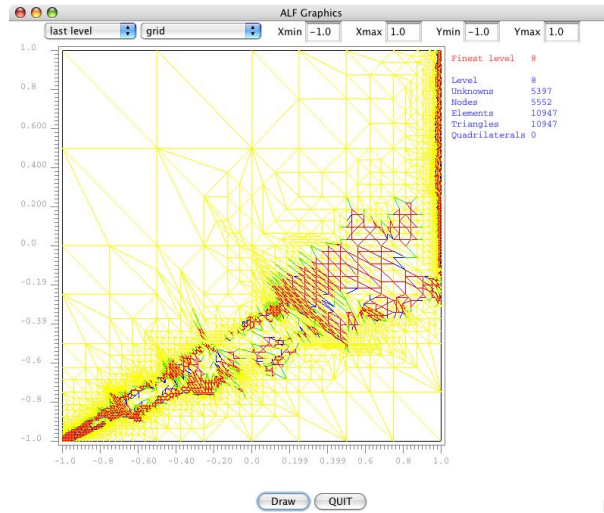


FIGURE 12. Mesh of Level 8 for a convection-diffusion equation

8. NUMBER OF NODES AND LEVELS

LF has two key-parameters:

- ML : the number of refinement levels and
- MN : the maximal number of nodes.

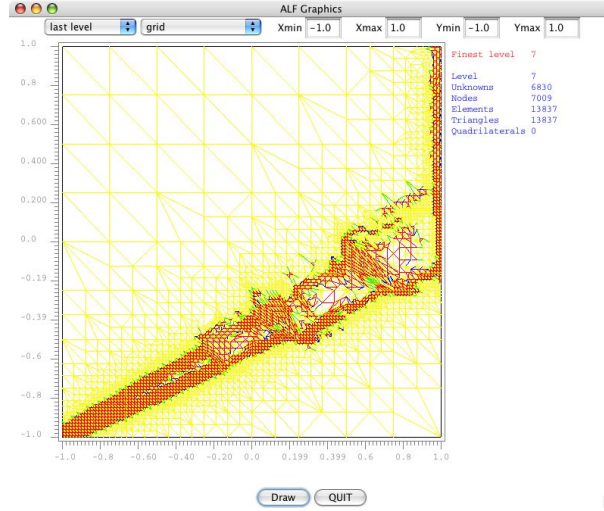


FIGURE 13. Mesh of Level 7 for a convection-diffusion equation

These are set in the text fields **Number of refinement levels** and **Maximal number of nodes** respectively.

The refinement-solution loop (cf. Section 7) is performed until the number ML of levels is reached, or the number MN of nodes is attained, or the allocated storage, which is a fixed factor of MN , is exhausted (cf. Section 10). Choosing a too small number of ML or MN naturally leads to computations with only a small number of grids that do not allow for an adequate finite element solution. Choosing a too large number of ML or – most critical – MN may cause troubles with the memory allocation of your Java virtual machine.

9. GRAPHICS

ALF's graphics window enables you to visualise the results of your computations.

In the top row of the window you'll see four text fields that determine the clipping area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. Usually you'll want to choose the values in these text fields such that the whole computational domain will be shown. But selecting other values allows you to get a zoom of sub-domains that might be of particular interest. *Note* that abscissa and ordinate scale separately, thus a circle will look like an ellipse if $x_{\max} - x_{\min}$ differs from $y_{\max} - y_{\min}$. Clicking the **Draw** button starts the graphics.

The left choice in the top row allows you to navigate through the various mesh-levels:

- *First level*: Go to the coarsest level.
- *Next level*: Increase the level count by 1.
- *Previous level*: Decrease the level count by 1.

- *Last level*: Go to the finest level.

The right choice in the top row lets you choose the type of graphics:

- *Grid*: Shows the grid of the actual level (cf. e.g. Figures 12 and 13).
- *Solution: isolines*: Shows the isolines of the computed solution on the actual grid (cf. e.g. Figure 14).

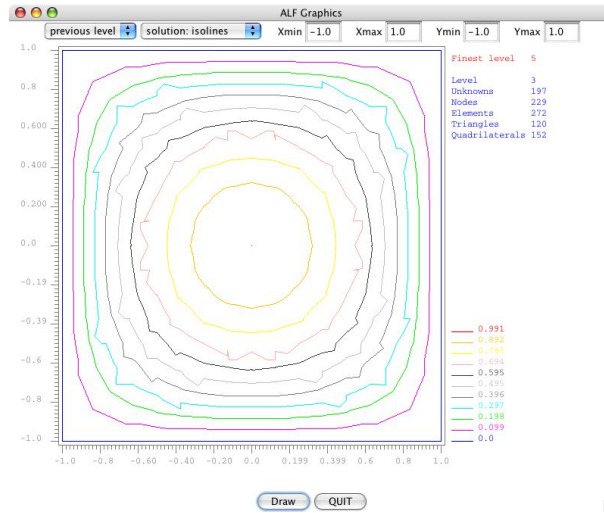


FIGURE 14. Isolines of a computed solution

- *Solution: colourplot*: Shows a colourplot of the computed solution on the actual grid (cf. Figure 15 for the same solution as in Figure 14).

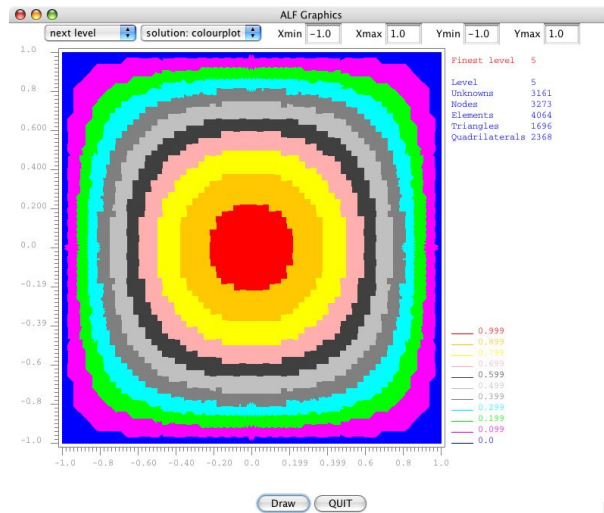


FIGURE 15. Colourplot of the finite element function of Fig. 14

- *Estimated error*: Shows the estimated errors. Each element is painted in a colour according to the relative size of the element's error indicator (cf. e.g. Figure 16). This option is only available in connection with an adaptive mesh refinement.

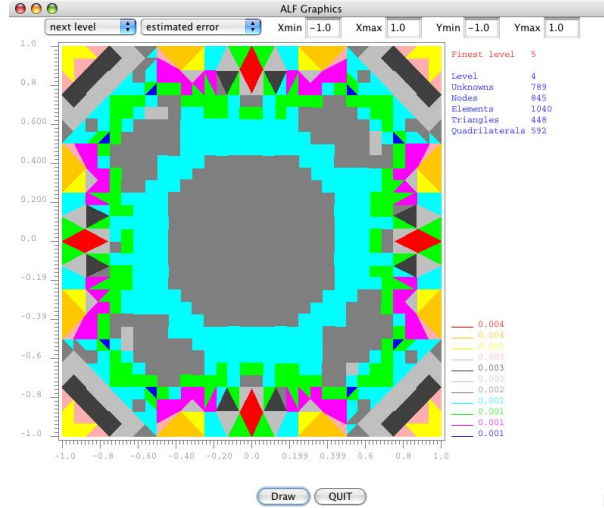


FIGURE 16. Visualisation of error indicators

10. ERROR HANDLING AND MEMORY REQUIREMENT

ALF stops its computations when either the prescribed number of refinement levels or the prescribed number of nodes is attained or when the allocated storage is exhausted. It then prints one of the following messages in the output area of its main window:

- *Final level reached*: A standard exit.
- *Too many new nodes and too many new elements*: A standard exit.
- *Too many new nodes*: A standard exit.
- *Too many new elements*: A standard exit.
- *Too many unknowns in stiffness matrix*: A less standard, but harmless exit.
- *Too many non-zero elements in stiffness matrix*: A less standard, but harmless exit.
- *Iterative solver did diverge*: In this case you should try another solution algorithm. When using a multigrid algorithm, increasing the number of smoothing steps and switching from a V- to a W-cycle might help.
- *No red elements found*: Indicates problems with the refinement strategy and should not appear.
- *Too small lv in stiffness matrix*: Should not appear.
- *Too large lv in stiffness matrix*: Should not appear.

Sometimes user-input data may be erroneous. ALF then prints an error message and stops the computation. In this case you may try to compute the same example with a new input or a completely new example. The following error messages may be encountered:

- *Inconsistent neighbour relation*: Indicates an erroneous user-input when using the options "user defined initial grid" or "user defined boundary, automatic tessellation".
- *Inconsistent node data*: Indicates an erroneous user-input when using the options "user defined initial grid" or "user defined boundary, automatic tessellation".
- *Inconsistent element data*: Indicates an erroneous user-input when using the options "user defined initial grid" or "user defined boundary, automatic tessellation".
- *No neighbour found for element ... with index at least ...*: Indicates an erroneous user-input when using the options "user defined initial grid" or "user defined boundary, automatic tessellation".
- *Too many nodes in first grid*: Should not appear.
- *Too many elements in first grid*: Should not appear.
- *Non-existing initial grid*: Should not appear.
- *No elements in first grid*: Should not appear.

When using a too large number of refinement levels or – more critical – a too large number of nodes, your Java virtual machine may run out of memory. It then throws an out of memory exception. ALF tries to catch this exception. In this case ALF prints the error message *Error: insufficient memory*. You then have to re-launch the applet.

If the error message did occur with your first example, you probably have chosen a too large a number of nodes. In this case you should enter a smaller number in the text field **Maximum number of nodes** and try again.

If the error message occurs after several successful computations, Java's garbage collector probably has not done its job. In this case you may try again with your old settings of the text field **Maximum number of nodes**.

ALF requires about $50MN$ doubles and $60MN$ integers where MN is the number given in the text field **Maximum number of nodes**.

RUHR-UNIVERSITÄT BOCHUM, FAKULTÄT FÜR MATHEMATIK, D-44780 BOCHUM, GERMANY

E-mail address: `rv@num1.rub.de`